CS-184: Computer Graphics

Lecture #19: Forward and Inverse Kinematics

Prof. James O'Brien
University of California, Berkeley

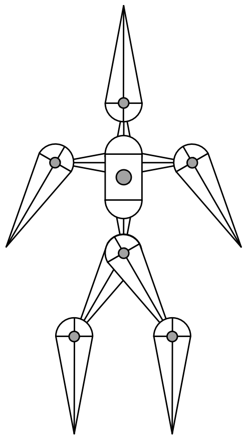
V2011-F-19-1.0

Today

- Forward kinematics
- Inverse kinematics
  - Pin joints
  - Ball joints
  - Prismatic joints

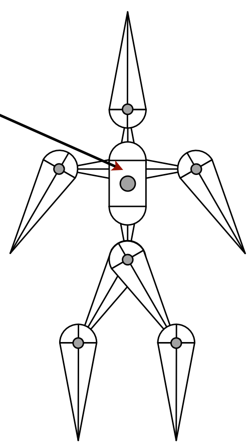
Forward Kinematics

- Articulated skeleton
  - Topology (what's connected to what)
  - Geometric relations from joints
  - Independent of display geometry
  - Tree structure
    - Loop joints break "tree-ness"



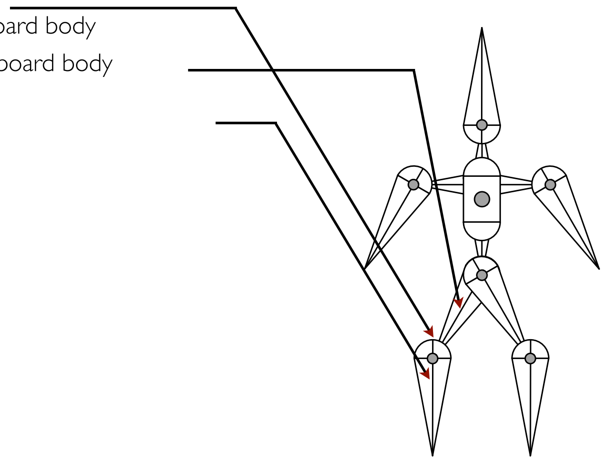
Forward Kinematics

- Root body
  - Position set by "global" transformation
  - Root joint
    - Position
    - Rotation
  - Other bodies relative to root
  - ***Inboard* toward the root**
  - ***Outboard* away from root**



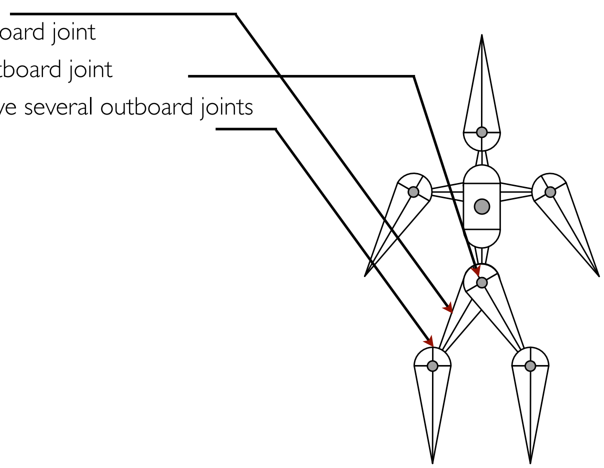
Forward Kinematics

- A joint
  - Joint's inboard body
  - Joint's outboard body



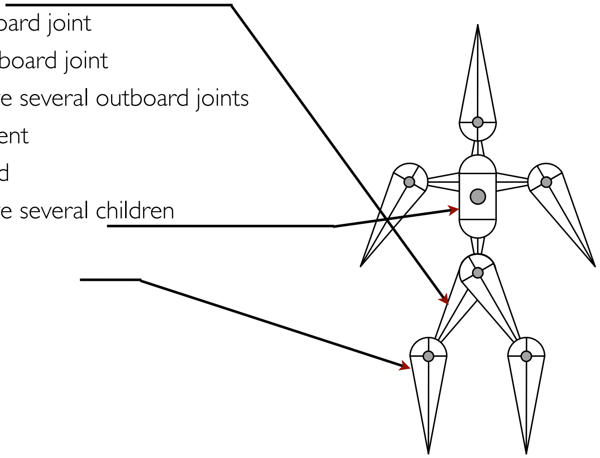
Forward Kinematics

- A body
  - Body's inboard joint
  - Body's outboard joint
    - May have several outboard joints



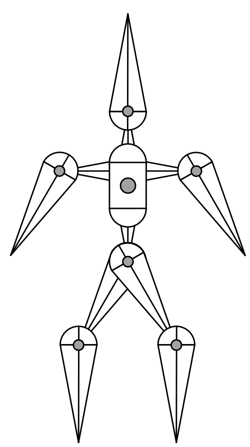
Forward Kinematics

- A body
  - Body's inboard joint
  - Body's outboard joint
    - May have several outboard joints
  - Body's parent
  - Body's child
    - May have several children



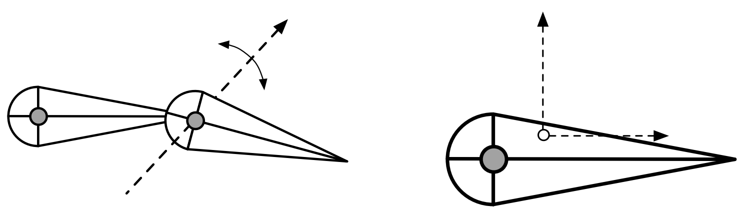
Forward Kinematics

- Interior joints
  - Typically not 6 DOF joints
  - Pin - rotate about one axis
  - Ball - arbitrary rotation
  - Prism - translation along one axis



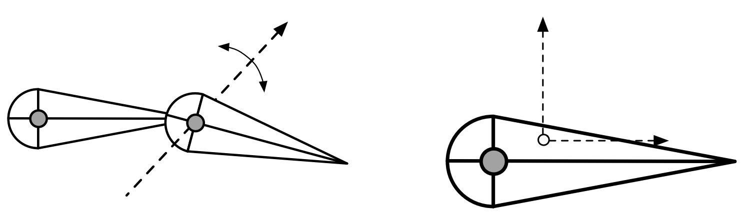
Forward Kinematics

- Pin Joints
  - Translate inboard joint to local origin
  - Apply rotation about axis
  - Translate origin to location of joint on outboard body



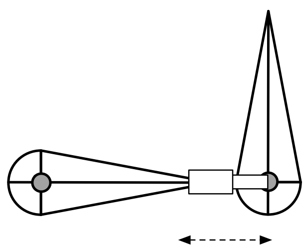
Forward Kinematics

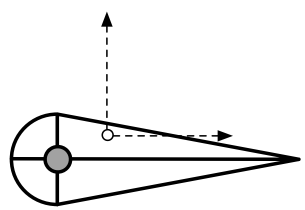
- Ball Joints
  - Translate inboard joint to local origin
  - Apply rotation about arbitrary axis
  - Translate origin to location of joint on outboard body



Forward Kinematics

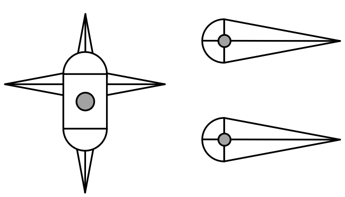
- Prismatic Joints
  - Translate inboard joint to local origin
  - Translate along axis
  - Translate origin to location of joint on outboard body





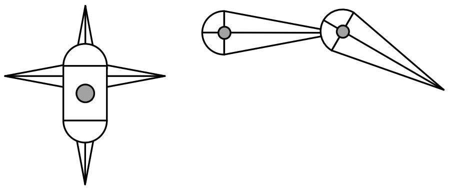
Forward Kinematics

- Composite transformations up the hierarchy



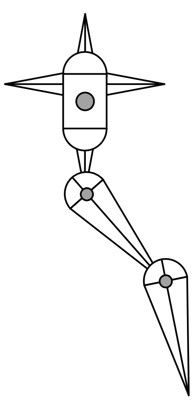
Forward Kinematics

- Composite transformations up the hierarchy



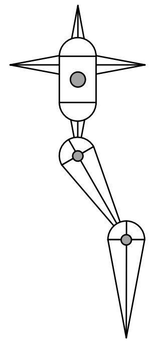
Forward Kinematics

- Composite transformations up the hierarchy



Forward Kinematics

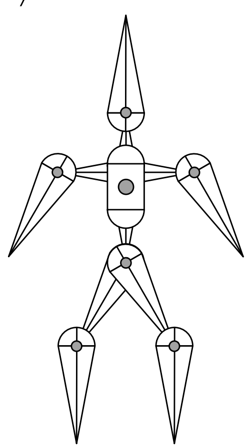
- Composite transformations up the hierarchy



15

Forward Kinematics

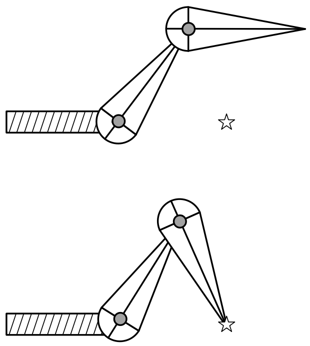
- Composite transformations up the hierarchy



16

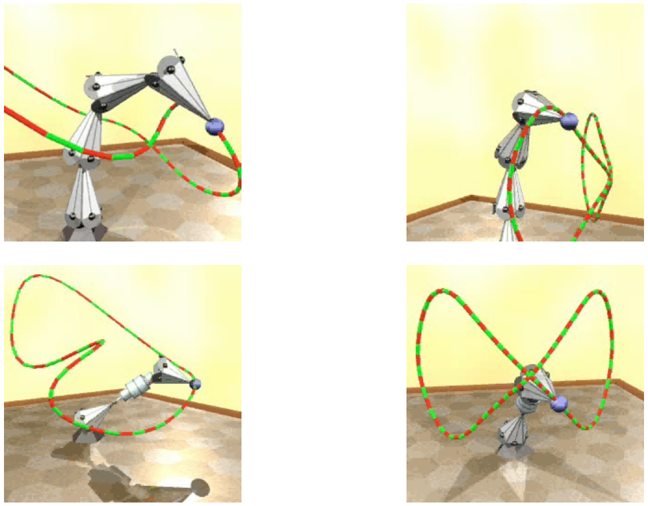
Inverse Kinematics

- Given
  - Root transformation
  - Initial configuration
  - Desired end point location
- Find
  - Interior parameter settings



17

Inverse Kinematics

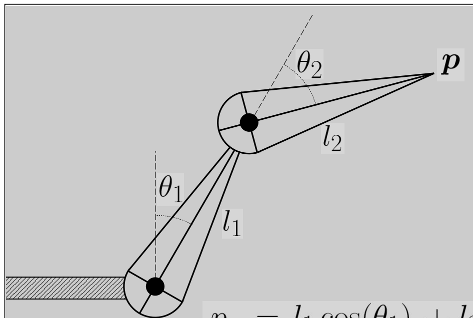


Egon Pasztor

18

Inverse Kinematics

- A simple two segment arm in 2D



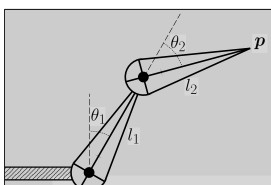
$$p_z = l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2)$$

$$p_x = l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2)$$

Warning: Z-up Coordinate System

Inverse Kinematics

- Direct IK: solve for the parameters

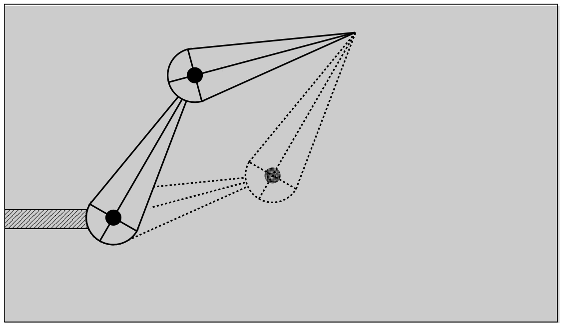


$$\theta_2 = \cos^{-1}\left(\frac{p_z^2 + p_x^2 - l_1^2 - l_2^2}{2 l_1 l_2}\right)$$

$$\theta_1 = \frac{-p_z l_2 \sin(\theta_2) + p_x(l_1 + l_2\cos(\theta_2))}{p_x l_2 \sin(\theta_2) + p_z(l_1 + l_2\cos(\theta_2))}$$

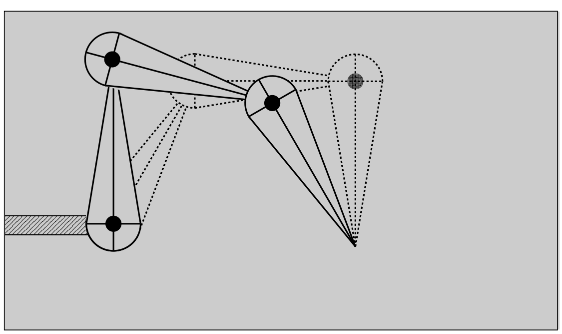
Inverse Kinematics

- Why is the problem hard?
  - Multiple solutions separated in configuration space



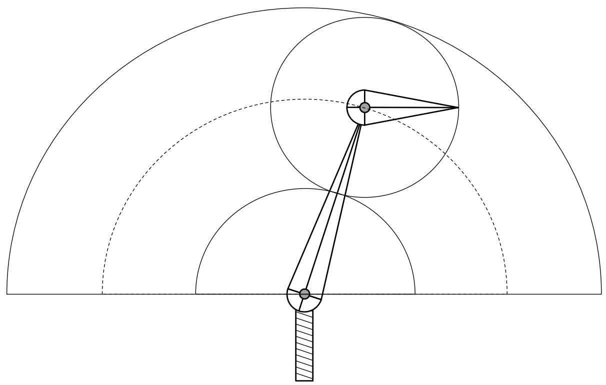
Inverse Kinematics

- Why is the problem hard?
  - Multiple solutions connected in configuration space



Inverse Kinematics

- Why is the problem hard?
  - Solutions may not always exist

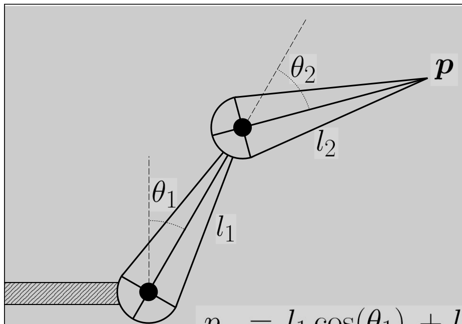


Inverse Kinematics

- Numerical Solution
  - Start in some initial configuration
  - Define an error metric (**e.g.** goal pos - current pos)
  - Compute Jacobian of error w.r.t. inputs
  - Apply Newton's method (or other procedure)
  - Iterate...

Inverse Kinematics

- Recall simple two segment arm:



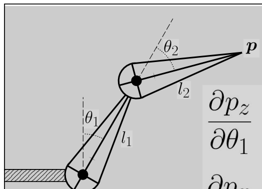
$$p_z = l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2)$$

$$p_x = l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2)$$

Warning: Z-up Coordinate System

Inverse Kinematics

- We can write of the derivatives



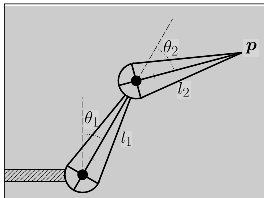
$$\frac{\partial p_z}{\partial \theta_1} = -l_1 \sin(\theta_1) - l_2 \sin(\theta_1 + \theta_2)$$

$$\frac{\partial p_x}{\partial \theta_1} = l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2)$$

$$\frac{\partial p_z}{\partial \theta_2} = - l_2 \sin(\theta_1 + \theta_2)$$

$$\frac{\partial p_x}{\partial \theta_2} = + l_2 \cos(\theta_1 + \theta_2)$$

Inverse Kinematics



Direction in Config. Space

$$\theta_1 = c_1\theta_*$$

$$\theta_2 = c_2\theta_*$$

$$\frac{\partial p_z}{\partial \theta_*} = c_1\frac{\partial p_z}{\partial \theta_1} + c_2\frac{\partial p_z}{\partial \theta_2}$$

Inverse Kinematics

The Jacobian (of p w.r.t. θ)

$$J_{ij} = \frac{\partial p_i}{\partial \theta_j}$$

Example for two segment arm

$$J = \begin{bmatrix} \frac{\partial p_z}{\partial \theta_1} & \frac{\partial p_z}{\partial \theta_2} \\ \frac{\partial p_x}{\partial \theta_1} & \frac{\partial p_x}{\partial \theta_2} \end{bmatrix}$$

Inverse Kinematics

The Jacobian (of p w.r.t. θ)

$$J = \begin{bmatrix} \frac{\partial p_z}{\partial \theta_1} & \frac{\partial p_z}{\partial \theta_2} \\ \frac{\partial p_x}{\partial \theta_1} & \frac{\partial p_x}{\partial \theta_2} \end{bmatrix}$$

$$\frac{\partial \boldsymbol{p}}{\partial \theta_*} = J \cdot \begin{bmatrix} \frac{\partial \theta_1}{\partial \theta_*} \\ \frac{\partial \theta_2}{\partial \theta_*} \end{bmatrix} = J \cdot \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}$$

Inverse Kinematics

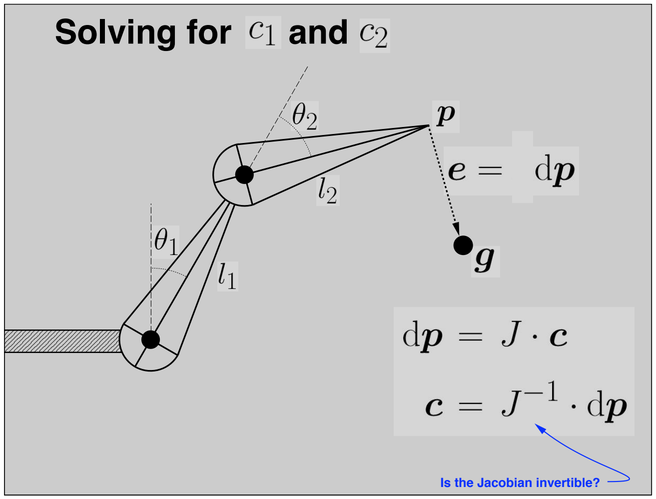
Solving for c_1 and c_2

$$\boldsymbol{c} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \qquad \mathrm{d}\boldsymbol{p} = \begin{bmatrix} \mathrm{d}p_z \\ \mathrm{d}p_x \end{bmatrix}$$

$$\mathrm{d}\boldsymbol{p} = J \cdot \boldsymbol{c}$$

$$\boldsymbol{c} = J^{-1} \cdot \mathrm{d}\boldsymbol{p}$$

Inverse Kinematics



Inverse Kinematics

- Problems
  - Jacobian may (will!) not always be invertible
    - Use pseudo inverse (SVD)
    - Robust iterative method
  - Jacobian is not constant
- Nonlinear optimization, but problem is (mostly) well behaved

$$J = \begin{bmatrix} \frac{\partial p_z}{\partial \theta_1} & \frac{\partial p_z}{\partial \theta_2} \\ \frac{\partial p_x}{\partial \theta_1} & \frac{\partial p_x}{\partial \theta_2} \end{bmatrix} = J(\theta)$$

Inverse Kinematics

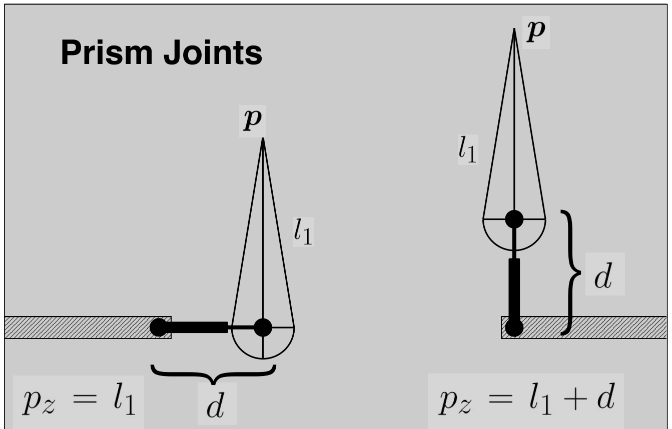
- More complex systems
  - More complex joints (prism and ball)
  - More links
  - Other criteria (COM or height)
  - Hard constraints (joint limits)
  - Multiple criteria and multiple chains

Inverse Kinematics

- Some issues
  - How to pick from multiple solutions?
  - Robustness when no solutions
  - Contradictory solutions
  - Smooth interpolation
    - Interpolation aware of constraints

Inverse Kinematics

Prism Joints



$p_z = l_1$

$p_x = d$

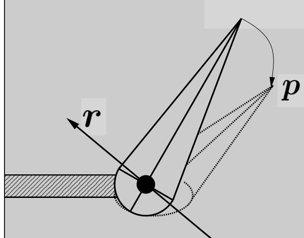
$p_z = l_1 + d$

$p_x = 0$

Inverse Kinematics

Ball Joints

$$
\begin{aligned}
\boldsymbol{p} &= \hat{\boldsymbol{r}}(\hat{\boldsymbol{r}} \cdot \boldsymbol{x}) \\
&+ \sin(||\boldsymbol{r}||)(\hat{\boldsymbol{r}} \times \boldsymbol{x}) \\
&- \cos(||\boldsymbol{r}||)(\hat{\boldsymbol{r}} \times (\hat{\boldsymbol{r}} \times \boldsymbol{x}))
\end{aligned}
$$



Inverse Kinematics

Ball Joints (moving axis)

$$\mathrm{d}\boldsymbol{p} = [\mathrm{d}\boldsymbol{r}]\cdot e^{[\boldsymbol{r}]}\cdot\boldsymbol{x} = [\mathrm{d}\boldsymbol{r}]\cdot\boldsymbol{p} = \underbrace{-[\boldsymbol{p}]}\cdot\mathrm{d}\boldsymbol{r}$$

That is the Jacobian for this joint

$$[\boldsymbol{r}] = \begin{bmatrix} 0 & -r_3 & r_2 \\ r_3 & 0 & -r_1 \\ -r_2 & r_1 & 0 \end{bmatrix}$$

$$[\boldsymbol{r}]\cdot\boldsymbol{x} = \boldsymbol{r}\times\boldsymbol{x}$$

Inverse Kinematics

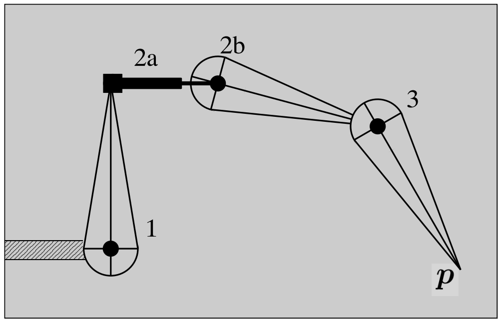
Ball Joints (fixed axis)

$$\mathrm{d}\boldsymbol{p} = (\mathrm{d}\theta)[\hat{\boldsymbol{r}}]\cdot\boldsymbol{x} = \underbrace{-[\boldsymbol{x}]\cdot\hat{\boldsymbol{r}}}\mathrm{d}\theta$$

That is the Jacobian for this joint

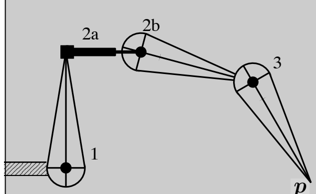
Inverse Kinematics

- Many links / joints
  - Need a generic method for building Jacobian



Inverse Kinematics

- Can't just concatenate individual matrices



~~$\tilde{J} = [J_3\ J_{2b}\ J_{2a}\ J_{1b}]$~~

$$\boldsymbol{d} = \begin{bmatrix} d_3 \\ d_{2b} \\ d_{2a} \\ d_{1b} \end{bmatrix}$$

$$\mathrm{d}\boldsymbol{p} \neq \tilde{J} \cdot \mathrm{d}\boldsymbol{d}$$

Inverse Kinematics

Transformation from body to world

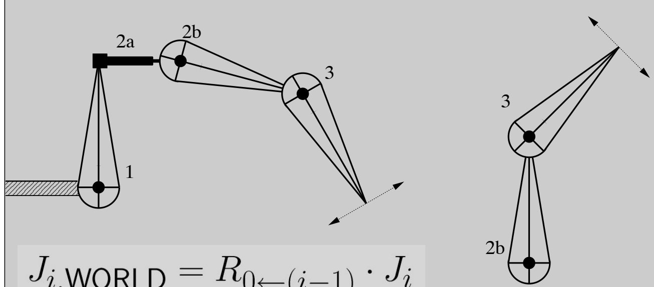
$$X_{0\leftarrow i} = \prod_{j=1}^{i} X_{(j-1)\leftarrow j} = X_{0\leftarrow 1} \cdot X_{1\leftarrow 2} \cdots$$

Rotation from body to world

$$R_{0\leftarrow i} = \prod_{j=1}^{i} R_{(j-1)\leftarrow j} = R_{0\leftarrow 1} \cdot R_{1\leftarrow 2} \cdots$$

Inverse Kinematics

Need to transform Jacobians to common coordinate system (WORLD)



$$J_{i,\text{WORLD}} = R_{0\leftarrow(i-1)} \cdot J_i$$

Inverse Kinematics

$$J = \begin{bmatrix} R_{0\leftarrow 2\mathrm{b}} \cdot & J_3(\theta_3, \boldsymbol{p_3}) \\ R_{0\leftarrow 2\mathrm{a}} \cdot & J_{2\mathrm{b}}(\theta_{2\mathrm{b}}, X_{2\mathrm{b}\leftarrow 3} \cdot \boldsymbol{p_3}) \\ R_{0\leftarrow 1} \cdot & J_{2\mathrm{a}}(\theta_{2\mathrm{a}}, X_{2\mathrm{a}\leftarrow 3} \cdot \boldsymbol{p_3}) \\ & J_1(\theta_1, X_{1\leftarrow 3} \cdot \boldsymbol{p_3}) \end{bmatrix}^T$$

Note: Each row in the above should be transposed....

$$\boldsymbol{d} = \begin{bmatrix} d_3 \\ d_{2\mathrm{b}} \\ d_{2\mathrm{a}} \\ d_{1\mathrm{b}} \end{bmatrix}$$

$$\mathrm{d}\boldsymbol{p} = J \cdot \mathrm{d}\boldsymbol{d}$$

Suggested Reading

- Advanced Animation and Rendering Techniques by Watt and Watt
  - Chapters 15 and 16